Real-Time Communication System Powered byAI for speciallyAbled

TEAM ID:PNT2022TMID00911

Team Members:

- Syed Wajith K - 211419104325
- Sharan SK - 211419104323
- Arunachalaeswar C - 211419104316
- Gugan RagavK - 211419104317

Industry Mentor(s) Name: Divya

Faculty Mentor(s) Name: Vinmathi M S

Project Report Format

1. **INTRODUCTION**
    a. Project Overview
    b. Purpose
2. **LITERATURE SURVEY**
    a. Existing problem
    b. References
    c. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
    a. Empathy Map Canvas
    b. Ideation & Brainstorming
    c. Proposed Solution
    d. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
    a. Functional requirement
    b. Non-Functional requirements
5. **PROJECT DESIGN**
    a. Data Flow Diagrams
    b. Solution & Technical Architecture
    c. User Stories
6. **PROJECT PLANNING & SCHEDULING**
    a. Sprint Planning & Estimation
    b. Sprint Delivery Schedule
    c. Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    a. Feature 1
    b. Feature 2
8. **TESTING**
    a. Test Cases
    b. User Acceptance Testing
9. **RESULTS**
    a. Performance Metrics
10. **ADVANTAGES & DISADVANTAGES**
11. **CONCLUSION & FUTURE SCOPE**
12. **APPENDIX**

    Source Code & GitHub & Project Demo Link

INTRODUCTION

1 INTRODUCTION

1.1 PROJECT OVERVIEW

In this sign language recognition project, we create a sign detector, which detects numbers and alphabets that can very easily be extended to cover a vast multitude of other signs and hand gestures including the alphabets.

Communication is very crucial to human beings, as it enables us to express ourselves. We communicate through speech, gestures, body language, reading, writing or through visual aids, speech being one of the most commonly used among them. However, unfortunately, for the speaking and hearing-impaired minority, there is a communication gap. Visual aids, or an interpreter, are used for communicating with them. However, these methods are rather cumbersome and expensive, and can't be used in an emergency. Sign Language chiefly uses manual communication to convey meaning. This involves simultaneously combining hand shapes, orientations and movement of the hands, arms or body to express the speaker's thoughts.

Sign Language consists of fingerspelling, which spells out words character by character, and word level association which involves hand gestures that convey the word meaning. Fingerspelling is a vital tool in sign language, as it enables the communication of names, addresses and other words that do not carry a meaning in word level association. In spite of this, fingerspelling is not widely used as it is challenging to understand and difficult to use. Moreover, there is no universal sign language and very few people know it, which makes it an inadequate alternative for communication.

A system for sign language recognition that classifies finger spelling can solve this problem. Various machine learning algorithms are used and their accuracies are recorded and compared in this report.

1.2 PROJECT PURPOSE

Speech impaired people use hand signs and gestures to communicate. Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing 2 and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

A functioning signing recognition system could provide a chance for the inattentive communication with non-signing people without the necessity for an interpreter. It might be wont to generate speech or text making the deaf more independent. Unfortunately, there has not been any system with these capabilities thus far. During this project our aim is to develop a system which may classify signing accurately.

Making a computer understand speech, facial expressions and human gestures are some steps towards it. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. The project aims to determine human gestures by creating an HCI. Coding of these gestures into machine language demands a complex programming algorithm. In our project we are focusing on Image Processing and Template matching for better output generation

LITERATURE SURVEY

2 LITERATURE SURVEY

2.1 EXISTING PROBLEM

Sign language is used for communication by community of hearing-impaired people. HI people can have impairments that vary from limited hearing to compete deafness. Since the process of learning to speak involves feedback from hearing what you say, people who are born with a hearing impairment are not able to properly speak. Therefore, they rely on other body parts to communicate. Hands for example are used to describe the shape of the object. Hands are also used to describe actions like “go from this place to that place by pointing first to source place and drawing a line visually in space toward the destination place”. In addition, facial expressions convey emotions. In this way body parts serves as alternative way of communication for hearing impaired people.

Sign making involves the upper part of human’s body. These body parts are categorized as manual and non-manual features. Manual features consist of hand’s configuration and according to four hand components are used in sign making: hand shape, hand orientation, hand movement and hand location. On the other side non-manual features consist of head movements, facial expressions and body postures Thus sign making involves both manual and non-manual features. Most of the meaning in sign language is conveyed through manual features, but understanding the full meaning of a sentence requires observation of non-manual features also. Non-manual features play an important role, especially in showing grammatical information The role of nonmanual features is elaborated in more detail in grammar’s section

2.2 REFERENCES

[1] Microsoft, “Kinect Fact Sheet,” 2010. [Online]. Available: www.microsoft.com/enus/news/presskits/xbox/docs/kinectfs.docx.

[2] W. C. Stoke, “Sign language structure: an outline of the visual communication systems

of the American deaf. 1960.,” J. Deaf Stud. Deaf Educ., vol. 10, no. 1, pp. 3–37, Jan.

2005.

[3] World Federation of Deaf, “Convention on the Rights of Persons with Disabilities - Sign

Language,” 2013. [Online]. Available: http://wfdeaf.org/human-rights/crpd/sign-language.

[4] Population Reference Bureau, “Population data sheet,” 2013.

[5] P. M. Lewis, G. F. Simons, and C. D. Fening, “Ethnologue: Languages of the world (17th

edition),” 2013. [Online]. Available: http://www.ethnologue.com/. [Accessed: 03-Dec2013].

2.3 PROBLEM STATEMENT

People with disabilities are not able to communicate with the people and society. Though technologies are evolving but there is no significant growth for these people. So, an AI system is developed to communicate with people in real time.

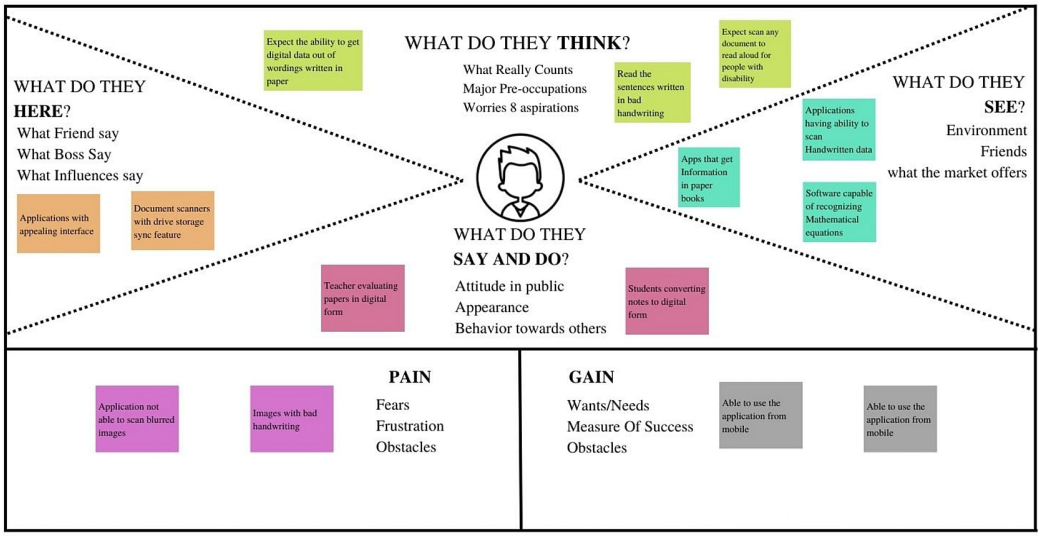
Dumb people use hand signs to communicate, hence normal people face problem in recognizing their language by signs made. Hence there is a need of the systems which recognizes the different signs and conveys the information to the normal people.

IDEATION & PROPOSED SOLUTION

3 IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

EMPATHY MAP



3.2 PROPOSED SOLUTION

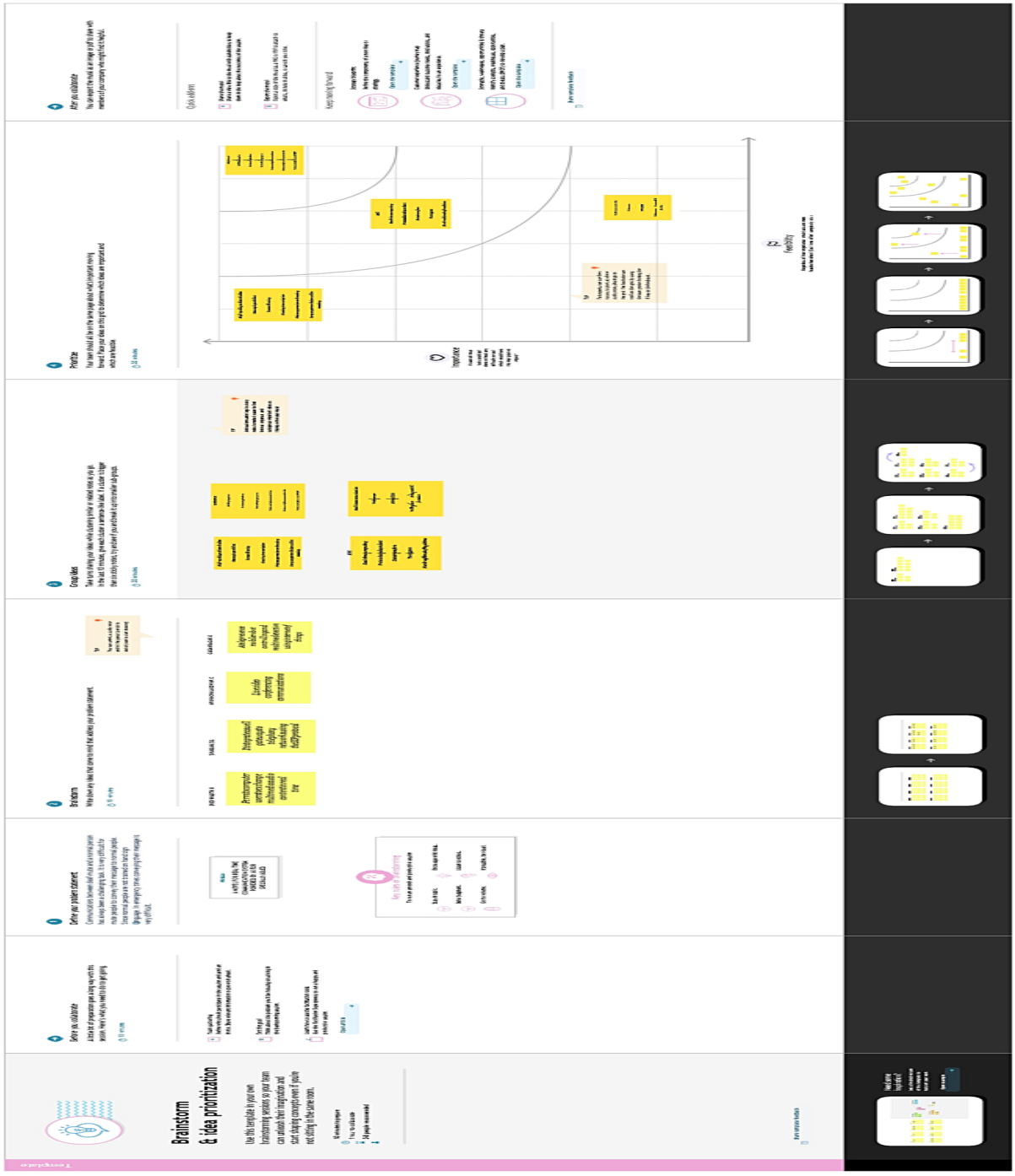
An application for deaf and dumb people to convey their information using signs which get converted to human-understandable languagean speech in Artificial Intelligence

By using Voice Conversion System with Hand Gesture Recognition and translation will be very useful to have a proper conversation

We are using a convolution neural network to create a model that is trainedon different hand gestures and an app is built for the use this mode

Communicating with others and being connected in the society and removeaccessibility barriers

3.3 IDEATION & BRAINSTORMING



3.4 PROBLEM SOLUTION FIT

REQUIREMENT ANALYSIS

4 REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

• System is presented as black box
• Hearing impaired is the person that performs the signs
• Normal hearing is the passive user of the system

The System Requirements Can Be Specified

1. Hearing impaired person should be able to perform sign that represent digit number
2. Hearing impaired person should be able to perform sign that represent alphabet letter 29
3. Hearing impaired person should be able to perform sign that represent word
4. Hearing impaired person should be able to perform sign that represent sentence
5. Hearing impaired person should be able to see the translation of sign to text
6. Hearing impaired person should be able to change the component (number/alphabet or word/sentence) for which translation to speech is provided

NORMAL FLOW

▪ User comes in front of camera and performs the alphabet letter
▪ System analyzes the performed sign
▪ System shows the sign meaning as text and speech

ALTERNATIVE FLOWS

- System shows that user is not detected
- User enters the field of view
- System shows that user is detected

4.2 NON-FUNCTIONAL REQUIREMENTS

The conditions on which system should operate are specified as non-functional requirements and they are:

Real time - the system should provide the recognition of signs and their translation to speech in an unnoticeable time for its users.

Accuracy – signs should not be confused and the system should recognize appropriate sign.

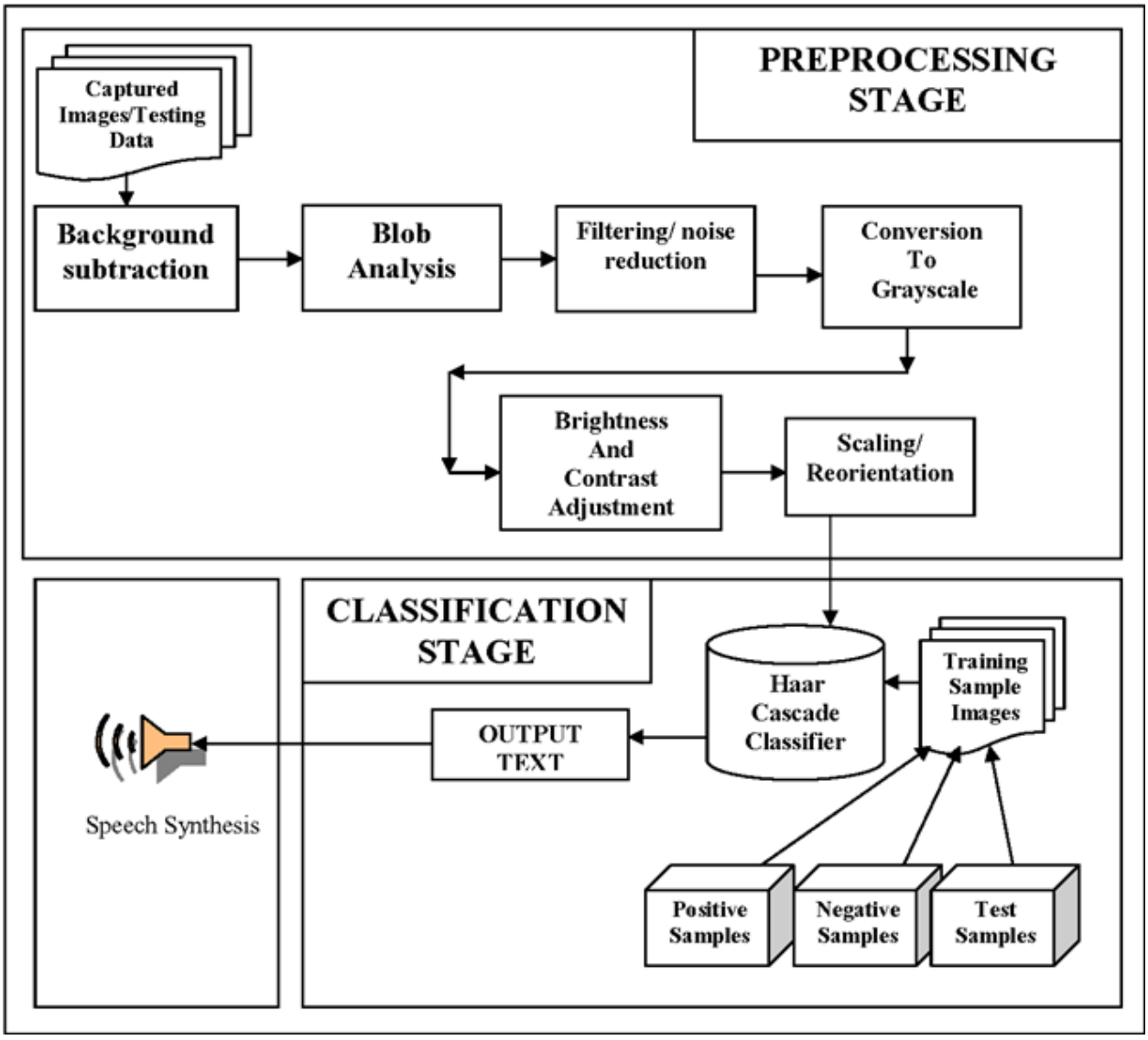
Environment – the system should provide real time recognition with high accuracy in low light conditions as well.

Usability – the system should provide natural interaction to its users. The hearing-impaired person needs to worry nothing else, just for performing signs.

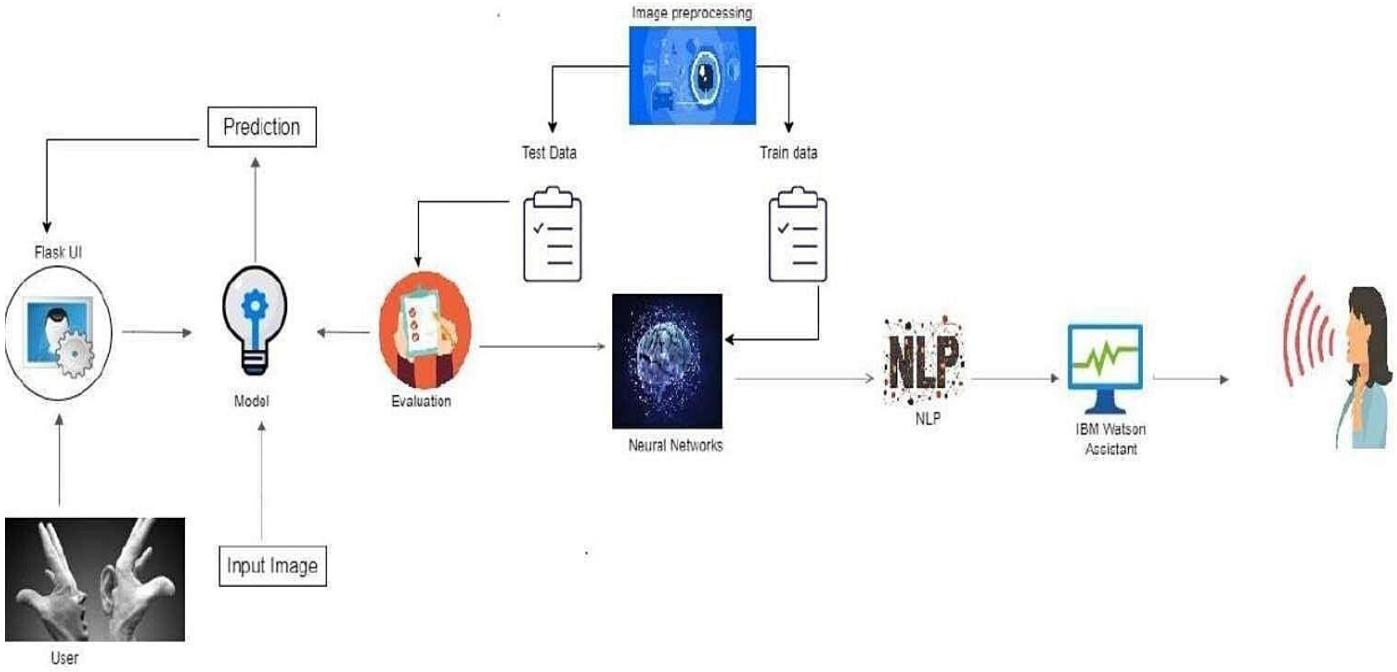
PROJECT DESIGN

5 PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming the password	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email &click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register via some third party's link	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can type manually and also can used saved login credentials	High	Sprint-1
	Dashboard	USN-6	As a customer, I can get all services and help through the dashboard	I can access my dashboard and change profile	Medium	Sprint-2
Customer (Web user)	Registration	USN-7	As a customer, I could able to login through registered phone number by using OTP instead of Gmail	I could able to register& login via phone number to access my account	High	Sprint-2
Customer Care Executive	Service	USN-8	Can avail the service by calling customer care or reaching through E-mail.	Can avail the service by calling customer care or reaching through E- mail.	Medium	Sprint-1
Administrator	Sign up	USN-9	Customer have to sign-up to use these things and all	Have to enter valid credentials.	High	Sprint-1
	Enrollment	USN-10	The customer can avail all services once he/she enrolled.	As customer it is quite enchanting.	Medium	Sprint-2

PROJECT PLANNING & SCHEDULING

6 PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

SPRINT	FUNCTIONAL REQUIREMENTS	USER STORY NUMBER	USER STORY/ TASK	STORY POINTS	PRIORITY	TEAM MEMBERS
Sprint -1	Registration	USN - 1	Collect Dataset	2	High	Syed Wajith K, Sharan S K, Arunachala Eswar C, Gugan Ragav K
Sprint -1	Login	USN - 2	Collecting key points using Media Pipe Holistic	1	High	Syed Wajith K Sharan S K
Sprint -2		USN - 3	Model Initialization With required Layers	1	Medium	Arunachala Eswar C, Gugan Ragav K
Sprint-2	Dashboard	USN - 4	As a user, I can log into my accountin a given Dashboard	1	High	Syed Wajith K, Gugan Ragav K
Sprint –1	User Interface	USN – 5	Professional responsible for user requirements & needs	1	High	Sharan S K, Arunachala Eswar C,
Sprint –3	Objective	USN - 6	The goal is to describe all the inputs & Outputs	1	High	Sharan S K Gugan Ragav K
Sprint –4	Privacy	USN - 7	The Developed applications .	1	High	Syed Wajith K, Arunachala Eswar C

6.2 SPRINT DELIVERY SCHEDULE

SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE(PLANNED)	STORY POINTS COMPLETED (AS ON PLANNED END DATE)	SPRINT RELEASE DATE
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint –2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint –3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

CODING & SOLUTIONING

7 CODING & SOLUTIONING

7.1 FEATURE 1

Importing The Required Model Building Libraries

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.utils import np_utils
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
dataset = pd.read_csv('E:\Datasets\Mall_Customers.csv')
```

Initializing The Model

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
spatial_dropout=0.05
recurrent_dropout=0.1
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

Adding The Convolution Layer

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
# let img1 be an image with no features
img1 = np.array([np.array([200, 200]), np.array([200, 200])])
img2 = np.array([np.array([200, 200]), np.array([0, 0])])
img3 = np.array([np.array([200, 0]), np.array([200, 0])])

kernel_horizontal = np.array([np.array([2, 2]), np.array([-2, -2])])
print(kernel_horizontal, 'is a kernel for detecting horizontal edges')

kernel_vertical = np.array([np.array([2, -2]), np.array([2, -2])])
print(kernel_vertical, 'is a kernel for detecting vertical edges')
```

In [ ]:
```
# We will apply the kernels on the images by
# elementwise multiplication followed by summation
def apply_kernel(img, kernel):
    return np.sum(np.multiply(img, kernel))

# Visualizing img1
plt.imshow(img1)
plt.axis('off')
plt.title('img1')
plt.show()

# Checking for horizontal and vertical features in image1
print('Horizontal edge confidence score:', apply_kernel(img1,
                                        kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img1,
                                        kernel_vertical))
```

In [ ]:
```
# Visualizing img2
plt.imshow(img2)
plt.axis('off')
plt.title('img2')
plt.show()

# Checking for horizontal and vertical features in image2
print('Horizontal edge confidence score:', apply_kernel(img2,
                                        kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img2,
                                        kernel_vertical))
```

In [ ]:
```
# Visualizing img3
plt.imshow(img3)
plt.axis('off')
plt.title('img3')
plt.show()

# Checking for horizontal and vertical features in image3
print('Horizontal edge confidence score:', apply_kernel(img3,
                                        kernel_horizontal))
print('Vertical edge confidence score:', apply_kernel(img3,
                                        kernel_vertical))
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

Adding The Pooling Layer

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
import numpy as np
from keras.models import Sequential
from keras.layers import MaxPooling2D
```

In [ ]:
```
# define input image
image = np.array([[[2, 2, 7, 3],
                   [9, 4, 6, 1],
                   [8, 5, 2, 4],
                   [3, 1, 2, 6]]])
image = image.reshape(1, 4, 4, 1)
```

In [ ]:
```
# define model containing just a single max pooling layer
model = Sequential(
        [MaxPooling2D(pool_size = 2, strides = 2)])

# generate pooled output
output = model.predict(image)
```

In [ ]:
```
# print output image
output = np.squeeze(output)
print(output)
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

Adding The Flatten Layer

In [ ]:
```
# importing numpy as np
import numpy as np
```

In [ ]:
```
# declare flatten np
gfg = np.array([[6, 9, 12], [8, 5, 2], [18, 21, 24]])

# using array.flatten() method
flat_gfg = gfg.flatten(order='A')
print(flat_gfg)
```

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

7.2 FEATURE 2

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
model = Sequential()
for i, feat in enumerate(args.conv_f):
    if i==0:
        model.add(Conv2D(feat, input_shape=x[0].shape, kernel_size=3, padding = 'same',use_bias=False))
    else:
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Conv2D(feat, kernel_size=3, padding = 'same',use_bias=False))
        model.add(BatchNormalization())
        model.add(LeakyReLU(alpha=args.conv_act))
        model.add(Dropout(args.conv_do[i]))
```

In [ ]:
```
model.add(Flatten())

#Input code here

denseArgs = {'use_bias':False}
for i,feat in enumerate(args.dense_f):
    model.add(Dense(feat,**denseArgs))
    model.add(BatchNormalization())
    model.add(LeakyReLU(alpha=args.dense_act))
    model.add(Dropout(args.dense_do[i]))
model.add(Dense(1))
```

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
model.add(Flatten())
```

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

Adding The Dense Layers

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
model.add(Dense(units=512, activation='relu'))
model.add(Dense(units=9, activation='softmax'))
```

In [ ]:
```
print("Adding dense layer on top")
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

In [ ]:
```
print("Complete architecture of the model")
model.summary()
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Compile To The Model

In [ ]:
```
from tensorflow.keras.preprocessing.image
import ImageDataGenerator
```

In [ ]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [ ]:
```
# Creating sample sourcecode to multiply two variables
# x and y.
srcCode = 'x = 10\ny = 20\nmul = x * y\nprint("mul =", mul)'

# Converting above source code to an executable
execCode = compile(srcCode, 'mulstring', 'exec')

# Running the executable code.
exec(execCode)
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
X_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
def compile_model_results(model, root="./"):

    listing = glob.glob(root + '/models/' + model + '/*/best_pars.pkl')

    dic_list = []
    for file in listing:
        tmp = hyper_parameters_load(file)
        dic_list.append(tmp.to_dictionary())

    df = pd.DataFrame(dic_list)
    df['diff'] = df.test_F1 - df.forecast_F1
    df['pci'] = abs(df.test_F1 - df.forecast_F1)

    if not os.path.exists(root + '/figures/' +  model ):
        os.makedirs(root + '/figures/' +  model )

    df.to_csv(root + '/figures/' +  model + '/results.csv', index=False)

    return df
```

In [ ]:
```
# Set optimizer loss and metrics
    opt = Adam(lr=args.initial_lr, beta_1=0.99, beta_2=0.999, decay=1e-6)
    if args.net.find('caps') != -1:
        metrics = {'out_seg': dice_hard}
    else:
        metrics = [dice_hard]

    loss, loss_weighting = get_loss(root=args.data_root_dir, split=args.split_num, net=args.net,
                                    recon_wei=args.recon_wei, choice=args.loss)

    # If using CPU or single GPU
    if args.gpus <= 1:
        uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
        return uncomp_model
    # If using multiple GPUs
    else:
        with tf.device("/cpu:0"):
            uncomp_model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)
            model = multi_gpu_model(uncomp_model, gpus=args.gpus)
            model.__setattr__('callback_model', uncomp_model)
        model.compile(optimizer=opt, loss=loss, loss_weights=loss_weighting, metrics=metrics)

X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Compilation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
```

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
# reading code from a file
f = open('main.py', 'r')
temp = f.read()
f.close()

code = compile(temp, 'main.py', 'exec')
exec(code)
```

Saving the Model

In [ ]:
```
model.save('asl_model_84_54.h5')
```

Fit And Save The Model

In [ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [ ]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/training_set',target_size=(64,64), class_mode='categorical',batch_size=900)
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set',target_size=(64,64), class_mode='categorical',batch_size=900)
```

```
Found 15760 images belonging to 9 classes.
Found 2250 images belonging to 9 classes.
```

In [ ]:
```
# Save Model Using Pickle
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
import pickle
```

In [ ]:
```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-
diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)
```

In [ ]:
```
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

In [ ]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  18
Len x-test :  3
```

In [ ]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[ ]:
```
{'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}
```

Model Creation

In [ ]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [ ]:
```
# Creating Model
model=Sequential()
```

In [ ]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
model.add(Flatten())
```

In [ ]:
```
# Adding Dense Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(9,activation='softmax'))
```

In [ ]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [ ]:
```
# Fitting the Model Generator
model.fit_generator(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future versio
n. Please use `Model.fit`, which supports generators.
```

```
Epoch 1/10
18/18 [==============================] - 92s 5s/step - loss: 0.0049 - accuracy: 0.9994 - val_loss: 0.2635 - val_accuracy: 0.9773
Epoch 2/10
18/18 [==============================] - 90s 5s/step - loss: 0.0040 - accuracy: 0.9995 - val_loss: 0.2074 - val_accuracy: 0.9773
Epoch 3/10
18/18 [==============================] - 87s 5s/step - loss: 0.0041 - accuracy: 0.9995 - val_loss: 0.2460 - val_accuracy: 0.9773
Epoch 4/10
18/18 [==============================] - 91s 5s/step - loss: 0.0041 - accuracy: 0.9992 - val_loss: 0.2470 - val_accuracy: 0.9782
Epoch 5/10
18/18 [==============================] - 88s 5s/step - loss: 0.0037 - accuracy: 0.9993 - val_loss: 0.2439 - val_accuracy: 0.9782
Epoch 6/10
18/18 [==============================] - 88s 5s/step - loss: 0.0024 - accuracy: 0.9997 - val_loss: 0.2852 - val_accuracy: 0.9782
Epoch 7/10
18/18 [==============================] - 91s 5s/step - loss: 0.0023 - accuracy: 0.9997 - val_loss: 0.2589 - val_accuracy: 0.9782
Epoch 8/10
18/18 [==============================] - 93s 5s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.2523 - val_accuracy: 0.9782
Epoch 9/10
18/18 [==============================] - 92s 5s/step - loss: 0.0013 - accuracy: 0.9999 - val_loss: 0.2269 - val_accuracy: 0.9778
Epoch 10/10
18/18 [==============================] - 91s 5s/step - loss: 0.0012 - accuracy: 0.9999 - val_loss: 0.2968 - val_accuracy: 0.9782
```

Out[ ]:

Saving the Model

In [ ]:
```
model.save('asl_model_84_54.h5')
```

TESTING

8 TESTING

8.1 TEST CASE

In [30]:
```
# Adding Layers
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

# Adding Hidden Layers
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

# Adding Output Layer
model.add(Dense(9,activation='softmax'))
```

In [31]:
```
# Compiling the Model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [32]:
```
# Fitting the Model Generator
model.fit(x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))
```

```
Epoch 1/10
6/6 [==============================] - 23s 4s/step - loss: 5.1206 - accuracy: 0.1690 - val_loss: 3.6585 - val_accuracy: 0.3119
Epoch 2/10
6/6 [==============================] - 22s 4s/step - loss: 2.3945 - accuracy: 0.3266 - val_loss: 1.5087 - val_accuracy: 0.4991
Epoch 3/10
6/6 [==============================] - 22s 4s/step - loss: 1.4384 - accuracy: 0.4037 - val_loss: 1.0430 - val_accuracy: 0.5836
Epoch 4/10
6/6 [==============================] - 23s 4s/step - loss: 1.0761 - accuracy: 0.6488 - val_loss: 0.7189 - val_accuracy: 0.7955
Epoch 5/10
6/6 [==============================] - 27s 5s/step - loss: 0.7835 - accuracy: 0.7774 - val_loss: 0.4046 - val_accuracy: 0.9501
Epoch 6/10
6/6 [==============================] - 25s 5s/step - loss: 0.5470 - accuracy: 0.8756 - val_loss: 0.2540 - val_accuracy: 0.9752
Epoch 7/10
6/6 [==============================] - 22s 4s/step - loss: 0.4018 - accuracy: 0.9090 - val_loss: 0.1675 - val_accuracy: 0.9799
Epoch 8/10
6/6 [==============================] - 22s 4s/step - loss: 0.2862 - accuracy: 0.9406 - val_loss: 0.1185 - val_accuracy: 0.9847
Epoch 9/10
6/6 [==============================] - 22s 4s/step - loss: 0.2108 - accuracy: 0.9612 - val_loss: 0.0880 - val_accuracy: 0.9863
Epoch 10/10
6/6 [==============================] - 22s 4s/step - loss: 0.1548 - accuracy: 0.9738 - val_loss: 0.0736 - val_accuracy: 0.9843
```

Out[32]:

Loading the Dataset & Image Data Generation

In [14]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [15]:
```
# Training Datagen
train_datagen = ImageDataGenerator(rescale=1/255,zoom_range=0.2,horizontal_flip=True,vertical_flip=False)
# Testing Datagen
test_datagen = ImageDataGenerator(rescale=1/255)
```

In [25]:
```
# Training Dataset
x_train=train_datagen.flow_from_directory(r'C:\Users\India\Desktop\Final_Project\Dataset\test_set',target_size=(64,64), class_mode='categorical',batch
# Testing Dataset
x_test=test_datagen.flow_from_directory(r'C:\Users\india\Desktop\Final_Project\Dataset\training_set',target_size=(64,64), class_mode='categorical',bat
```

```
Found 4969 images belonging to 9 classes.
Found 4969 images belonging to 9 classes.
```

In [26]:
```
print("Len x-train : ", len(x_train))
print("Len x-test : ", len(x_test))
```

```
Len x-train :  6
Len x-test :  6
```

In [27]:
```
# The Class Indices in Training Dataset
x_train.class_indices
```

Out[27]: {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H': 7, 'I': 8}

Model Creation

In [28]:
```
# Importing Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [29]:
```
# Creating Model
model=Sequential()
```

8.3 USER ACCPETANCE TESING

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	2	3	2	18
Duplicate	1	3	4	0	8
External	3	5	0	0	8
Fixed	12	2	5	22	41
Not Reproduced	0	1	0	0	1
Skipped	0	0	1	2	3
Won't Fix	0	4	1	1	7
Totals	27	17	14	27	86

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	49	0	0	49
Security	4	0	0	4
Outsource Shipping	4	0	0	4
Exception Reporting	11	0	0	11
Final Report Output	2	0	0	2
Version Control	1	0	0	1

RESULTS

9 RESULTS

9.1 Performance Metrics

Technical Skills Evaluation Metrics

S.No	Metrics	Weightage (%)	Description	Sub Evaluation Metrics & Scoring Criteria	Score	Description	Standard Template	Scoring Guidelines
1	Technical Training & Assignments	[illegible]	[illegible]	Number of Tests / Quizzes Attempted (Total - 4 Nos.) Assignment-1 Assignment-2 Assignment-3 Assignment-4	2 2 2 2	[illegible] Weekly Assignment Weekly Assignment Weekly Assignment Weekly Assignment	No	[illegible] Completed - 2, Partial - 1, Not Submitted - 0
2	Innovation & Problem Solving	[illegible]	[illegible] 1. Define the Problem Statement 2. Empathize & Discover 3. Brainstorm & Prioritize Ideas	1. Empathize & Discover	[illegible]	[illegible]	Yes	Completed - 2, Partial - 1, Not Submitted - 0
				2. Define the Problem Statement	[illegible]	[illegible]	Yes	[illegible]
				3. Brainstorm & Prioritize Ideas	[illegible]	[illegible]	Yes	[illegible]
3	Requirement Analysis using Critical Thinking	[illegible]	[illegible]	Determine the Requirements (Customer Journey Maps)	[illegible]	[illegible]	Yes	[illegible]
				Requirement Analysis (Functional, Operational, Technical) / Flow Charts	[illegible]	[illegible]	Yes	[illegible]
				Interpretation & Modeling	[illegible]	[illegible]	No	[illegible]
4	Project Design using Design Thinking	[illegible]	[illegible] 1. Proposed solution - Novelty - Feasibility of Concept - Social / Business Impact 2. Problem - Solution Fit 3. Solution Architecture	Proposed solution - Novelty - Feasibility of Concept - Social / Business Impact	[illegible]	[illegible]	Yes	[illegible]
				Problem - Solution Fit	[illegible]	[illegible]	Yes	[illegible]
				Solution Architecture	[illegible]	[illegible]	Yes	[illegible]
5	Technology Stack	[illegible]	[illegible]	Technical Architecture: - Is the system robust - Is it highly modifiable? - Is it scalable? - Is it buildable (w.r.t time & budget constraints)	[illegible]	[illegible]	No	[illegible]
				Open Source Frameworks	[illegible]	[illegible]	No	[illegible]
				Third-party API's	[illegible]	[illegible]	No	[illegible]
				Cloud Deployment	[illegible]	[illegible]	No	[illegible]
6	Project Planning using Agile Methodologies	[illegible]	[illegible] 1. Project Milestones & Tasks 2. Sprint Delivery Schedule 3. Distribution of Tasks among team members 4. Team management tool for agile planning	Project Milestones & Tasks	[illegible]	[illegible]	Yes	[illegible]
				Sprint Delivery Plan	[illegible]	[illegible]	Yes	[illegible]
				Project Progress Tracking	[illegible]	[illegible]	No	[illegible]
				Team management tool for agile planning (e.g. Trello, Jira etc.)	[illegible]	[illegible]	No	[illegible]
7	Coding & Solutioning	[illegible]	[illegible]	No. of Functional Features Included in the solution	[illegible]	[illegible]	No	[illegible]
				Code Layout, Readability and Reusability	[illegible]	[illegible]	No	[illegible]
				Utilization of Algorithms, Dynamic Programming, Optimal Memory Utilization	[illegible]	[illegible]	No	[illegible]
				Debugging & Traceability	[illegible]	[illegible]	No	[illegible]
				Exception Handling	[illegible]	[illegible]	No	[illegible]
8	Acceptance Testing	[illegible]	[illegible]	UAT Initiation	[illegible]	[illegible]	No	[illegible]
				UAT Design	[illegible]	[illegible]	No	[illegible]
				UAT Execution	[illegible]	[illegible]	No	[illegible]
				UAT Report Submission	[illegible]	[illegible]	Yes	[illegible]
				Utilization of Test management Tools	[illegible]	[illegible]	No	[illegible]
9	Performance Testing	[illegible]	[illegible]	Application Performance Metrics [illegible]	[illegible]	[illegible]	Yes	[illegible]
				Model Performance Metrics [illegible]	[illegible]	[illegible]	Yes	[illegible]
				Dashboard & User Story Metrics [illegible]	[illegible]	[illegible]	Yes	[illegible]

ADVANTAGES & DISADVANTAGES

10 ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

- It reduces frustration
- It increases self esteem
- It enhances language and listening skills
- It enriches relationships
- It increases their IQ

10.2 DISADVANTAGES

- A denial is possible.
- Legal validity lacking.
- A non-suitability clause allows
- I am amazed at the level of interpretation
- The presence of emotion within.
- By presenting an irrelevant speech.

CONCLUSION & FUTURE SCOPE

11 CONCLUSION & FUTURE SCOPE

11.1 CONCLUSION

The aim of this project is to predict the ISL alphanumeric hand-gestures in real time. The above work shows that it can be solved with better accuracy when we actually consider the segmented RGB hand-gestures. By applying depth-based segmentation we remove the overheads of dynamic background.

The segmented RGB hand-gestures were fed to 3 layered CNN for training and testing in real time. We were able to achieve training accuracy of 89.30% and testing accuracy of 98.5%. Our model showed good accuracy while predicting results both offline and online.

Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them.

The system didn’t perform too well but it was demonstrated that it can be built a first-person sign language translation system using only cameras and convolutional neural networks.

This project aims to predict sign language recognition using machine vision with the help of deep learning. The
performance of this tool is on part with that of humans for distinguishing the sign language gesture with real-time image.

.

11.2 FUTURE SCOPE

In the future we tend to help the autistic children with the need of real information as a classroom environment to test the actual scenario. After we collect the autistic children's real data, future work can be carried out.

The current research mainly identifies the emotions of the person, but the teaching process is a one-to-one interaction process.

If teachers' behaviours are included in the scope of recognition, it will be more realistic.

Then after recognizing the emotional performance of autistic children, we can consider evaluating the score for the emotions expressed and the result will be displayed.

If the score is less, it will motivate the person to practice more. In addition, we will include anxiety, surprise, and much more emotions.

We can develop a model for ISL word and sentence level recognition. This will require a system that can detect changes with respect to the temporal space.

We can develop a complete product that will help the speech and hearing-impaired people, and thereby reduce the communication gap.

APPENDIX

12 APPENDIX

12.1 SOURCE CODE

```python
import cv2

video = cv2.VideoCapture(0)

while True:
    ret, frame = video.read()
    cv2.imshow("Frame", frame)
    k = cv2.waitKey(1)
    if k == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

```python
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

class Video(object):
    def __init__(self):
        self.video = cv2.VideoCapture(0)
        self.roi_start = (50, 150)
        self.roi_end = (250, 350)
        self.model = load_model('asl_model.h5') # Execute Local Trained Model
        # self.model = load_model('IBM_Communication_Model.h5') # Execute IBM Trained Model
        self.index=['A','B','C','D','E','F','G','H','I']
        self.y = None
    def __del__(self):
        self.video.release()
    def get_frame(self):
        ret,frame = self.video.read()
        frame = cv2.resize(frame, (640, 480))
        copy = frame.copy()
        copy = copy[150:150+200,50:50+200]
        # Prediction Start
        cv2.imwrite('image.jpg',copy)
        copy_img = image.load_img('image.jpg', target_size=(64,64))
        x = image.img_to_array(copy_img)
        x = np.expand_dims(x, axis=0)
        pred = np.argmax(self.model.predict(x), axis=1)
        self.y = pred[0]
        cv2.putText(frame,'The Predicted Alphabet is: '+str(self.index[self.y]),(100,50),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,0),3)
        ret,jpg = cv2.imencode('.jpg', frame)
        return jpg.tobytes()
```

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

/* Set a style for all buttons */
button {
  background-color: #273298;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}

button:hover {
  opacity: 0.8;
}
```

```
/* Extra styles for the cancel button */
.cancelbtn {
  width: auto;
  padding: 10px 18px;
  background-color: #f44336;
}

/* Center the image and position the close button */
.imgcontainer {
  text-align: center;
  margin: 24px 0 12px 0;
  position: relative;
}

img.avatar {
  width: 40%;
  border-radius: 50%;
}

.container {
  padding: 16px;
}

span.psw {
  float: right;
  padding-top: 16px;
}

/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
```

```
    overflow: auto; /* Enable scroll if needed */
    background-color: rgb(0,0,0); /* Fallback color */
    background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
    padding-top: 60px;
  }

  /* Modal Content/Box */
  .modal-content {
    background-color: #fefefe;
    margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
    border: 1px solid #888;
    width: 80%; /* Could be more or less, depending on screen size */
  }

  /* The Close Button (x) */
  .close {
    position: absolute;
    right: 25px;
    top: 0;
    color: #000;
    font-size: 35px;
    font-weight: bold;
  }

  .close:hover,
  .close:focus {
    color: red;
    cursor: pointer;
  }

  /* Add Zoom Animation */
  .animate {
    -webkit-animation: animatezoom 0.6s;
    animation: animatezoom 0.6s
  }
```

```
@-webkit-keyframes animatezoom {
  from {-webkit-transform: scale(0)}
  to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
  from {transform: scale(0)}
  to {transform: scale(1)}
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}
</style>
</head>
<body>

<h2 style="text-align: center; color: rgba(1, 2, 2, 0.874);">REAL TIME COMMUNICATION SYSTEM POWERED BY AI FOR SPECIALLY ABLED</h2>

<button onclick="document.getElementById('id01').style.display='block'">Login</button>

<div id="id01" class="modal">

  <form class="modal-content animate" action="/action_page.php" method="post">
    <div class="imgcontainer">
      <span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close Modal">&times;</span>
      <img src="https://static.vecteezy.com/system/resources/thumbnails/007/407/996/small/user-icon-person-icon-client-symbol-login-head-sign-icon-design-vector.jpg">
    </div>
```

```
    <div class="container">
      <label for="uname"><b>Username</b></label>
      <input type="text" placeholder="Enter Username" name="uname" required>

      <label for="psw"><b>Password</b></label>
      <input type="password" placeholder="Enter Password" name="psw" required>

      <button type="submit">Login</button>
      <label>
        <input type="checkbox" checked="checked" name="remember"> Remember me
      </label>
    </div>

    <div class="container" style="background-color:#f1f1f1">
      <button type="button" onclick="document.getElementById('id01').style.display='none'" class="cancelbtn">Cancel</button>
      <span class="psw">Forgot <a href="#">password?</a></span>
    </div>
  </form>
</div>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>
<div class="display-cover">
    <video autoplay></video>
    <canvas class="d-none"></canvas>

    <div class="video-options">
        <select name="" id="" class="custom-select">
            <option value="">Select camera</option>
        </select>
    </div>

    <img class="screenshot-image d-none" alt="">

    <div class="controls">
        <button class="btn btn-danger play" title="Play"><i data-feather="play-circle"></i></button>
        <button class="btn btn-info pause d-none" title="Pause"><i data-feather="pause"></i></button>
        <button class="btn btn-outline-success screenshot d-none" title="ScreenShot"><i data-feather="image"></i></button>
    </div>
</div>

<script src="https://unpkg.com/feather-icons"></script>
<script src="script.js"></script>
</body>
<html><head>
</head><body>
    <video src="" ></video>
    <br />
<button id='flipCamera'>Flip</button>
</body>
<script>
  var front = false;
var video = document.querySelector('video');
  document.getElementById('flipCamera').onclick = function() { front = !front; };
  var constraints = { video: { facingMode: (front? "user" : "environment"), width: 640, height: 480  } };
  navigator.mediaDevices.getUserMedia(constraints)
  .then(function(mediaStream) {
    video.srcObject = mediaStream;
    video.onloadedmetadata = function(e) {
    video.play();
};
})
.catch(function(err) { console.log(err.name + ": " + err.message); })
</script></html>
</html>
```

```
<style>
.screenshot-image {
    width: 150px;
    height: 90px;
    border-radius: 4px;
    border: 2px solid whitesmoke;
    box-shadow: 0 1px 2px 0 rgba(0, 0, 0, 0.1);
    position: absolute;
    bottom: 5px;
    left: 10px;
    background: white;
}

.display-cover {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 70%;
    margin: 5% auto;
    position: relative;
}

video {
    width: 100%;
    background: rgba(0, 0, 0, 0.2);
}

.video-options {
    position: absolute;
    left: 20px;
    top: 30px;
}

.controls {
    position: absolute;
    right: 20px;
    top: 20px;
```

```
        display: flex;
    }

    .controls > button {
        width: 45px;
        height: 45px;
        text-align: center;
        border-radius: 100%;
        margin: 0 6px;
        background: transparent;
    }

    .controls > button:hover svg {
        color: white !important;
    }

    @media (min-width: 300px) and (max-width: 400px) {
        .controls {
            flex-direction: column;
        }

        .controls button {
            margin: 5px 0 !important;
        }
    }

    .controls > button > svg {
        height: 20px;
        width: 18px;
        text-align: center;
        margin: 0 auto;
        padding: 0;
    }

    .controls button:nth-child(1) {
        border: 2px solid #1a12b3;
    }
```

```
.controls button:nth-child(1) svg {
    color: #2b128e;
}

.controls button:nth-child(2) {
    border: 2px solid #008496;
}

.controls button:nth-child(2) svg {
    color: #008496;
}

.controls button:nth-child(3) {
    border: 2px solid #0048b5;
}

.controls button:nth-child(3) svg {
    color: #0f0a5b;
}

.controls > button {
    width: 45px;
    height: 45px;
    text-align: center;
    border-radius: 100%;
    margin: 0 6px;
    background: transparent;
}

.controls > button:hover svg {
    color: rgb(75, 173, 230);
}
</style>

<script>
// Get the modal
var modal = document.getElementById('id01');
```

```
// When the user clicks anywhere outside of the modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
feather.replace();

const controls = document.querySelector('.controls');
const cameraOptions = document.querySelector('.video-options>select');
const video = document.querySelector('video');
const canvas = document.querySelector('canvas');
const screenshotImage = document.querySelector('img');
const buttons = [...controls.querySelectorAll('button')];
let streamStarted = false;

const [play, pause, screenshot] = buttons;

const constraints = {
  video: {
    width: {
      min: 1280,
      ideal: 1920,
      max: 2560,
    },
    height: {
      min: 720,
      ideal: 1080,
      max: 1440
    },
  }
};
</script>
<script>
const getCameraSelection = async () => {
  const devices = await navigator.mediaDevices.enumerateDevices();
```

```
    const videoDevices = devices.filter(device => device.kind === 'videoinput');
    const options = videoDevices.map(videoDevice => {
      return `<option value="${videoDevice.deviceId}">${videoDevice.label}</option>`;
    });
    cameraOptions.innerHTML = options.join('');
  };

  </script>
  <script>

  play.onclick = () => {
    if (streamStarted) {
      video.play();
      play.classList.add('d-none');
      pause.classList.remove('d-none');
      return;
    }
    if ('mediaDevices' in navigator && navigator.mediaDevices.getUserMedia) {
      const updatedConstraints = {
        ...constraints,
        deviceId: {
          exact: cameraOptions.value
        }
      };
      startStream(updatedConstraints);
    }
  };

  const startStream = async (constraints) => {
    const stream = await navigator.mediaDevices.getUserMedia(constraints);
    handleStream(stream);
  };

  const handleStream = (stream) => {
    video.srcObject = stream;
    play.classList.add('d-none');
    pause.classList.remove('d-none');
```

```
    screenshot.classList.remove('d-none');
    streamStarted = true;
  };

  getCameraSelection();
  ...
  cameraOptions.onchange = () => {
    const updatedConstraints = {
      ...constraints,
      deviceId: {
        exact: cameraOptions.value
      }
    };
    startStream(updatedConstraints);
  };

  const pauseStream = () => {
    video.pause();
    play.classList.remove('d-none');
    pause.classList.add('d-none');
  };

  const doScreenshot = () => {
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0);
    screenshotImage.src = canvas.toDataURL('image/webp');
    screenshotImage.classList.remove('d-none');
  };

  pause.onclick = pauseStream;
  screenshot.onclick = doScreenshot;
  </script>

  </body>
  </html>
```

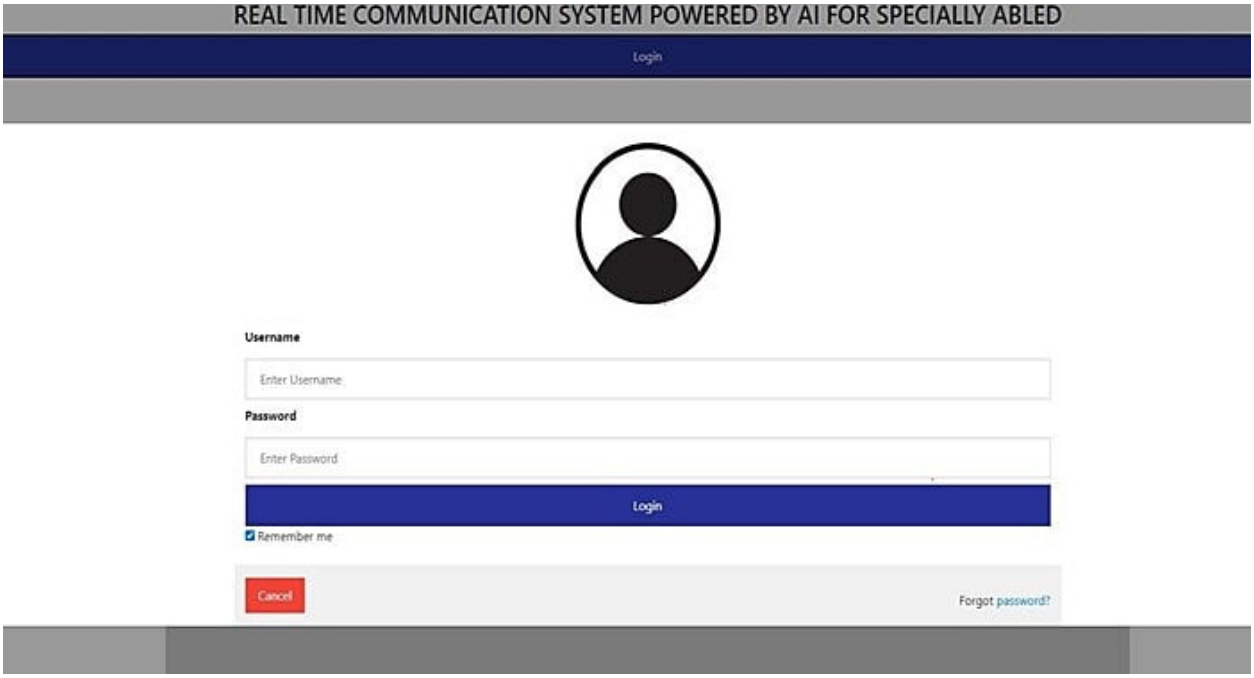
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>SmartBridge_WebApp_VideoTemplate</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.12.0/css/all.css">
    <link rel="stylesheet" href="assets/css/Banner-Heading-Image.css">
    <link rel="stylesheet" href="assets/css/Navbar-Centered-Brand.css">
    <link rel="stylesheet" href="assets/css/styles.css">
</head>

<body style="background: rgb(39,43,48);">
    <nav class="navbar navbar-light navbar-expand-md py-3" style="background: #212529;">
        <div class="container">
            <div></div><a class="navbar-brand d-flex align-items-center" href="#"><span
                    class="bs-icon-sm bs-icon-rounded bs-icon-primary d-flex justify-content-center align-items-center me-2 bs-icon"><i
                        class="fas fa-flask"></i></span><span style="color: rgb(255,255,255);">Real-Time Communication
                    System Powered By AI For Specially Abled</span></a>
            <div></div>
        </div>
    </nav>
    <section>
        <div class="d-flex flex-column justify-content-center align-items-center">
            <div class="d-flex flex-column justify-content-center align-items-center" id="div-video-feed"
                style="width: 640px;height: 480px;margin: 10px;min-height: 480px;min-width: 640px;border-radius: 10px;border: 4px dashed rgb(255,255,255) ;">
                <img src="{{ url_for('video_feed') }}" style="width: 100%;height: 100%;color: rgb(255,255,255);text-align: center;font-size: 20px;"
                    alt="Camera Access Not Provided!">
            </div>
        </div>
        <div class="d-flex flex-column justify-content-center align-items-center" style="margin-bottom: 10px;"><button
                class="btn btn-info" type="button" data-bs-target="#modal-1" data-bs-toggle="modal">Quick Reference
                -<strong> ASL Alphabets</strong></button></div>
    </section>
    <section>
```

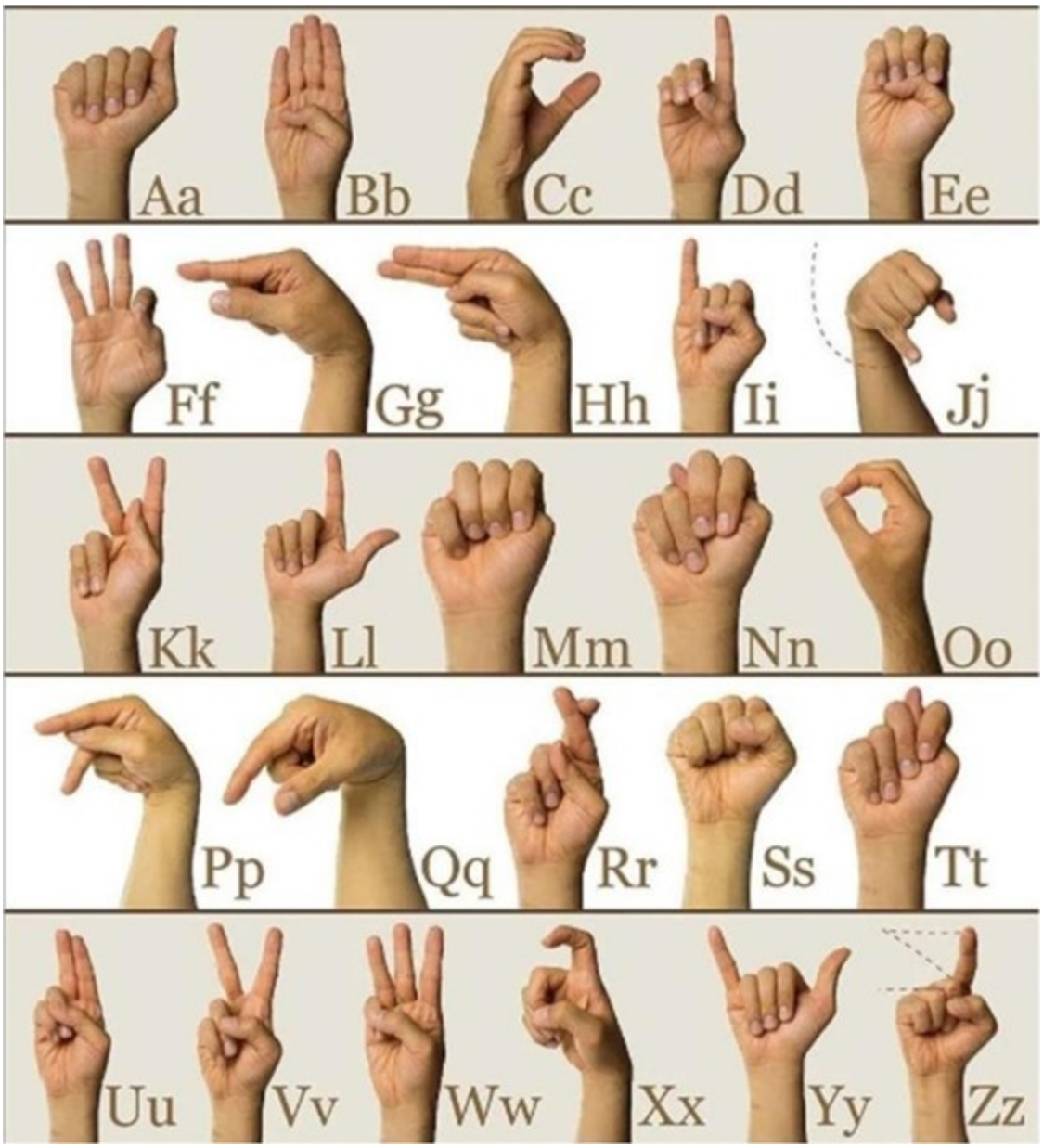
```
        <div class="container">
            <div class="accordion text-white" role="tablist" id="accordion-1">
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-button" data-bs-toggle="collapse"
                            data-bs-target="#accordion-1 .item-1" aria-expanded="true"
                            aria-controls="accordion-1 .item-1"
                            style="background: rgb(39,43,48);color: rgb(255,255,255);">About The Project</button></h2>
                    <div class="accordion-collapse collapse show item-1" role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Artificial Intelligence has made it possible to handle our daily activities
                                in new and simpler ways. With the ability to automate tasks that normally require human
                                intelligence, such as speech and voice recognition, visual perception, predictive text
                                functionality, decision-making, and a variety of other tasks, AI can assist people with
                                disabilities by significantly improving their ability to get around and participate in
                                daily activities.<br><br>Currently, Sign Recognition is available <strong>only for
                                    alphabets A-I</strong> and not for J-Z, since J-Z alphabets also require Gesture
                                Recognition for them to be able to be predicted correctly to a certain degree of
                                accuracy.</p>
                        </div>
                    </div>
                </div>
                <div class="accordion-item" style="background: rgb(33,37,41);">
                    <h2 class="accordion-header" role="tab"><button class="accordion-button collapsed"
                            data-bs-toggle="collapse" data-bs-target="#accordion-1 .item-2" aria-expanded="false"
                            aria-controls="accordion-1 .item-2"
                            style="background: rgb(39,43,48);color: rgb(231,241,255);">Developed By</button></h2>
                    <div class="accordion-collapse collapse item-2" role="tabpanel" data-bs-parent="#accordion-1">
                        <div class="accordion-body">
                            <p class="mb-0">Students at VIT-Bhopal University during SmartBridge AI Externship
                                Program.<br><br>1. <strong>Nirlov Deb</strong> 19BCG10067<br>2.
                                <strong>Kushagra</strong> 19BCG10025<br>3. <strong>Kartik Dhasmana</strong> 19BCG10002
                            </p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
```

```
    </section>
    <div class="modal fade" role="dialog" tabindex="-1" id="modal-1">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">American Sign Language - Alphabets</h4><button type="button"
                        class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
                </div>
                <div class="modal-body"><img src="{{ url_for('static', filename='img/ASL_Alphabets.png') }}" width="100%"></div>
                <div class="modal-footer"><button class="btn btn-secondary" type="button"
                        data-bs-dismiss="modal">Close</button></div>
            </div>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
</body>

</html>
```

```
.bs-icon {
  --bs-icon-size: .75rem;
  display: flex;
  flex-shrink: 0;
  justify-content: center;
  align-items: center;
  font-size: var(--bs-icon-size);
  width: calc(var(--bs-icon-size) * 2);
  height: calc(var(--bs-icon-size) * 2);
  color: var(--bs-primary);
}

.bs-icon-xs {
  --bs-icon-size: 1rem;
  width: calc(var(--bs-icon-size) * 1.5);
  height: calc(var(--bs-icon-size) * 1.5);
}

.bs-icon-sm {
  --bs-icon-size: 1rem;
}

.bs-icon-md {
  --bs-icon-size: 1.5rem;
}

.bs-icon-lg {
  --bs-icon-size: 2rem;
}

.bs-icon-xl {
  --bs-icon-size: 2.5rem;
}

.bs-icon.bs-icon-primary {
  color: var(--bs-white);
  background: var(--bs-primary);
```

```
38  }
39
40  .bs-icon.bs-icon-primary-light {
41    color: var(--bs-primary);
42    background: rgba(var(--bs-primary-rgb), .2);
43  }
44
45  .bs-icon.bs-icon-semi-white {
46    color: var(--bs-primary);
47    background: rgba(255, 255, 255, .5);
48  }
49
50  .bs-icon.bs-icon-rounded {
51    border-radius: .5rem;
52  }
53
54  .bs-icon.bs-icon-circle {
55    border-radius: 50%;
56  }
```



12.2 GITHUB & PROJECT DEMO LINK

https://github.com/IBM-EPBL/IBM-Project-3834-1658648678

https://drive.google.com/file/d/1k_WV0-1K8nPpTTd6SG6YEFystinbgL8r/view?usp=share_link